Generalized chart constraints for efficient PCFG and TAG parsing: Supplementary Materials

Stefan Grünewald and **Sophie Henning** and **Alexander Koller**
Department of Language Science and Technology
Saarland University, Saarbrücken, Germany
{stefang|shenning|koller}@coli.uni-saarland.de

A Training details

Both neural networks were implemented using Tensorflow 1.1.0.

A.1 Chart constraints

The network has two hidden layers consisting of Tensorflow LSTM cells with 100 units each. Weights are initialized by sampling from a uniform probability distribution with values between -0.1 and 0.1. No dropout is applied between layers.

As input, the network uses 100-dimensional pretrained word embeddings and a one-hot encoding for POS tags. Word embeddings for unknown words (UNK) and numbers (NUMBER) were initialized using a random normal distribution with a standard deviation of 0.5. Input sentences are processed one-by-one, i.e. no batching is performed.

We used the RMSProp optimizer for training, with a starting learning rate of $5 \cdot 10^{-4}$. The learning rate was decreased by 10% after each training epoch. The training process was stopped after 6 epochs, when accuracy on the development set stopped increasing. On an AMD Opteron 6380 processor with a clock rate of 2.5 GHz, the training process took about 4 hours in total. Tagging the entire test set takes about 10 seconds.

A.2 Supertagging

The network has two hidden layers consisting of Tensorflow LSTM cells. The first layer consists of 200 units, the second layer of 100 units. Weights are initialized by sampling from a uniform probability distribution with values between -0.1 and 0.1. A dropout of 50% is applied between layers during training.

As input, the network uses 200-dimensional pretrained word embeddings. Word embeddings for unknown words (UNK) and numbers (NUMBER) were initialized using a random normal distribution with a standard deviation of 0.5. The network does not use POS tags as input. Input sentences are processed one-by-one, i.e. no batching is performed.

We used the Adam optimizer for training, with a starting learning rate of $5 \cdot 10^{-4}$. The learning rate was decreased by 10% after each training epoch. The training process was stopped after 6 epochs, when accuracy on the development set stopped increasing. On an AMD Opteron 6380 processor with a clock rate of 2.5 GHz, the training process took about 11 hours in total. Tagging the entire test set takes about 10 seconds.